BETA BASIC NEWSLETTER No. 2

First, I must humbly apologise too many of you subscribers who will receive this issue late. (Since the Newsletter doesn't really get out-of-date, future subscribers will receive this issue as soon as they take out a subscription.) This was because I took on a contract programming project which had a shorter deadline than I really wanted, and penalty clauses. As tends to happen, it all took about twice as long as one might reasonably expect, and the Newsletter and some correspondence and support services (such as relocating printer drivers) have been delayed. I am determined that this will not happen again; the third Newsletter will be published after a fairly short time-lag and even early subscribers should end up getting 6 issue within a year.

Continuing with the apologies, I must admit that the listings in issue 1 were not proof-read - there did not seem to be any need, since they were tested and than transferred direct to Tasword by a simple program. Unfortunately, the program was a little too simple, since it did not cope will lines OF more than 61 characters, leading to corruption of line 50 of the Nowotnik puzzle, which should read:

```
50 FOR n=1 TO 8
     PRINT AT 2+n,8; PAPER 2;a$; PAPER 6;a$; AT 10+n,8;
     PAPER 4;a$; PAPER 5;93
   NEXT n
```

The same problem struck line 120 of PRCC grab, which should be:

```
120 LET a$=MEMORY$()(16384+third*2048 TO18431+third*2048)
     +MEMORY$()(22528+thrid*256 TO 22783+third*256)
```

And on the last page, PROC keyprint should have in line 8110:

```
       OR a$=CHR$ 12 OR a$=CHR$ 13
```

Many thanks to Daniel Ben-Sefer, Israel, for pointing these out. (My first reaction was "what's wrong with the rest of them?" - which is pretty hypocritical, considering the fact that ay usual reaction to a bug in commercial software is to think "well, it's not too bad - and they must already know about it". But please, do COMPLAIN!)

One further problem: some of you will find that the FOR-NEXT loop designed to demonstrate PROC grab prints 64 letters in the same place, rather than filling a third of the screen. This is due to a bug fix which corrected a minor problem with a WINDOW example in the manual. The "fix" has now been fixed, but if you have a "problem" copy of Beta Basic, you can get round the problem by using a CSIZE other than 0 as the initial CSIZE - e.g. enter CSIZE 2 as a direct command - then run line 10 exactly as listed.

Since I mentioned the professions of some subscribers in the first issue, a number of readers have written with interesting personal details. W.E. Thomson is now retired, but his first program, -_0 years ago, was for the Ferranti "Pegasus" computer. Charles Buszard is Secretary of the Royal Photographic Society. Graham Todd is Mayor of Crawley.

****

PROC hide - making procedures invisible.

There are two flaws in the claim that procedures provide new commands exactly like "real" commands; they appear in your listings, and they are not safe from NEW. PROC hide corrects this by renumbering any lines before line 100 to zero. Beta Panic will accept any number of line zero's and keep them out of sight unless you use LIST D. The lines are safe from NEW.

You cannot use a GOTO or GOSUB within the hidden DEF PROCs, and in any case PROC hide does not change anything except the initial line :umber. Besides, you shouldn't need to use GOTO or GOSUB!

```
100 DEF PROC hide
      LET x=DPEEK (23635)
      DO
        LET x=x+DPEEK (x+2) +4
      EXIT IF PEEK (x+1)=100
        POKE x+1,0
      LOOP
    END PROC
```

The System a variable PROM (at 23635) is used to find the start of the program, and then the internal information on line number and length is used to renumber all lines up to line 100 as 0. Here are two trivial procedures to try it out on:

```
10 DEF PROC invert
      BORDER 0
      PAPER 0
      INK 7
      ALTER TO INK, 7, PAPER 0
    END PROC

20 DEF PRCC pip
      BEEP .1,1
    END PROC
```

Typing (space)hide will make the two procedures vanish. If you use NEW, the commands PIP and INVERT will still work. You can SAVE the program (perhaps after MERGEing the Beta Basic loader) and re-LOAD which your, new commands intact.

****

PROC show - revealing invisible procedures.

If you have used PROC hide and then wish to edit your invisible lines, you will need PROC show. Any line zeros after BB's own line zero are renumbered as etc.

```
200  DEF PROC show
       LET x=DPEEK(23635),num=1
       DO
         LET x=x+DPEEK(x+2)+4
       EXIT IF DPEEK(x)<>0
         POKE x+1,num
         LET num=num+1
       LOOP
     END PROC
```

****

USING MACHINE CODE WITH BETA BASIC.

I am not sure how many BB users actually write machine cede, ,but judging from our correspondence, the use of small subroutines is quite common. One problem sometimes experienced is illustrated by a user who uses a home-made printer interface: on lowering RAMTOP and inserting his printer driving software below BB's coda, he found everything worked C. K. till he tried using WINDOW ERASE, which crashed the machine. This happened because Beta Basic uses a zero byte as a marker at the end of DEF KEY and WINDOW definitions, ,which are stored above RAMTOP. If this is over-written, then some DEF KEY and WINDOW commands will not work correctly.

The Beta Basic 1.8 manual warned that this marker should be left alone, but with the improved CLEAR of BB 3.0, this was emitted. The reasoning was that if someone needed 50 bytes of space, they would use CLEAR 50, which would make 50 bytes of space below BB's code by moving any DEF KEYs and WINDOW definitions (and their end marker) down by 50 bytes. Unfortunately the end marker looks very like unused memory and tends to get used as a 51st. byte! It would have been better to use something other than zero as an end marker.

I thought it would be useful to go through a complete example of the use of a machine code subroutine with Beta Basic, in this case an improved sound command. We will put the routine just below BB's code. Exactly where that is varies according to which version of the program you have got, since bugs like that mentioned on page 1 require some memory to correct. If you bought BB before about March 86, RAMTOP on first loading will be 47070 (if it is 47270, you have a *very* early tape and should exchange it) and BB's code starts at 47072. Later versions may be different. To check, MERGE the Basic loader and then PRINT rt (=RAMTOP). BB's code will always start at the initial RAMTOP, plus 2. I'll assume RAMTOP started at 47070 - correct the addresses if you need to. The sound routine code takes 36 bytes. CLEAR 36 opens a space which ends at 47071, and starts 35 bytes lower, at 47036. (1 sometimes get confused by the fact that a block of code which has start and end addresses differing by, say, I byte, has a *length* of 2 bytes.) Below are the lines to open the space and load the machine code. If "ERROR!" is printed, check and correct the DATA statement, delete line 10 and RUN again.

```
10 CLEAR 36
20 LET t=0
30 FOR n=47036 TO 47071
     READ x
     LET t=t+x
     POKE n,x
   NEXT n
40 IF t<>5696 THEN PRINT "ERROR!"
50 DATA 243,237,91,248,255,42,250,255,237,75,252,255,197,213,
   229,17,1,0,205,181,3,225,209,193,205,84,31,2O8,9,27,122,179
   ,32,234,251,201
```

Now, how do we get the code to do anything? It requires some values to be POKEd into memory before being executed by RANDOMIZE USR 47036. The next section is really a digression -skip it if you like, and go on to PROC sound.

****

DO WE NEED A CALL COMMAND?

Some users haves suggested a CALL command in Beta Basic. At its simplest, such a command simply replaces the often odd looking requirement use a command before USR. The PROC below does just that:

```
10 DEF PPOC call adr
20   RANDOMIZE USR adr
30 END PROC
```

It can easily la improved to accept parameters after the CALL, either number or string, which can be placed in known locations (I have used the last character pattern in the UDG area) to be used by the machine code, e.g. for 2 numbers:

```
40 DEF PROC call adr,byte1,byte2
50   POKE 65528,byte1
60   POKE 65529,byte2
70   RANDOMIZE USR adr
80 END PROC
```

or a string:

```
 90 DEF PROC call adr,a$
100  DPOKE 65528,LENGTH(0,"a$")
110  DPOKE 65530,LEN a$
120  RANDOMIZE USR adr
130 END PROC
```

(In line 100, LENGTH is used to find the address of the string's first character.) This is all very well - but you still have to make sure you type the address correctly every time; e.g. CALL 47036 is hardly a user-friendly command. In most cases, you would want to write a procedure which gave you a friendlier "front-end" to the machine code - like the one below:

****

PROC sound - sound effects with variable duration, pitch and rise/fall rate.

To use this procedure, you have to enter the machine code part given on the previous page. The first parameter is the number of cycles in the sound, the second is the initial duration of each cycle (pitch) and the third (optional) parameter is the amount to change the pitch by after each cycle. The time the sound actually lasts is related to all 3 parameters! Sometimes people ask why Beta Basic doesn't have an improved sound command. Partly this was because I couldn't imagine documenting such a 'messy" command - I wanted something that lasted for x seconds and went up or down by selectable octaves-per-second. This is quite tricky to write. However, I notice even the QL has a sound command which requires you to just throw in some numbers and see what happens, so in retrospect perhaps I was wrong to be so fussy.

```
100 FOR n=1 TO 10
      PRINT n
110   sound 30,1000,n
120 NEXT n
```

```
200 DEF PROC sound leng,note,rise
        DEFAULT rise=0
        DPOKE 65523,leng
        DPOKE 65530,note
        DPOKE 65532,65536*(rise>0)-rise
        RANDOMIZE USR 47036
      END PROC
```

Notes:

1. The odd-looking final DPOKE allows negative to be used with the last parameter, i.e. the pitch can fall.

2. If the cycle length drops to zero or less, the sound will become a dull clicking, rather than a high note.

3. BREAK will stop a sound at any point.

****

A NOTE ON DISC SYSTEMS.

Beta Basic is available in special versions for the OPUS Discovery system and the Wafadrive. These versions support disc or wafa as DEFAULT SAVE/LOAD device, as well as the built-in printer ports, EOF, and ON ERROR. However, each version required more than a week of programming effort, and we are reluctant to do this for every possible system, particularly if the manufacturers keep altering their product in ways that cause problems. The cost of the hardware is also a factor. (Anyone want to buy a slightly used Wafadrive?)

A partial solution is provided by the use of procedures. The example below is for the Technology Research Beta Disc, but it may get people with other systems thinking. The Beta Disc requires the following beautiful syntax for SAVE:

```
RANDOMIZE USR 15363: REM : SAVE "name"
```

However, if you have a procedure such as:

```
DEF PROC sv a$
   RANDOMIZE USR 15363
   REM
   SAVE a$ END PROC
```

you can simply type:

```
(space)sv "name"
```

which is a lot easier. The same principle can be applied to LOAD, MERGE, VERIFY, CAT, and entry to DOS. The complete set of procedures can be concealed by PROC hide (in this issue). (Beta Disc users should note that Beta Basic is "turned off" by LOAD and MERGE from disc and must be "turned on" again by RANDOMIZE USR 53419.)

****

PROC ellipses - ellipse drawing with specified size, angle and position on.

PROC polygon - polygon drawing with specified side number, and position.

(A polygon is a symmetrical multi-sided shape like a triangle, square or pentagon)

This impressive contribution is from Steve Drain of Birmingham. Both PROC ellipse and PROC polygon use a general-purpose PROC Poly but With different entry parameters. The list of parameters for PROC ellipse looks pretty horrible, but then you need to specify radius along the long axis (m) and the short axis - (n),and you might also wish to set the angle of the long axis in degrees from horizontal, (a) and the position of the centre (x and y). Accuracy is controlled by the variable s. Since the default angle is zero, and the default position is the centre of the screen you can type just: ellipse 80,30 to get an ellipse with a long axis of 80 and a short axis of 30. Frankly, the math's makes me scratch my head (I'll stick to hexadecimal) PROC ellipse works at a respectable speed by using a series of straight lines to provide a good approximation of a true ellipse. The number of lines used is controlled by the variable s.

PROC polygon uses its first parameter, p, to select the number of sides on the polygon; essentially, it draws an "ellipse" with both axes equal (a circle!) with terrible accuracy by using a small value for the number of lines to use! If p=3 you get a triangle, 4 is a square, and so on. The second parameter (m) c controls size, (a) controls angle, and x and y give position. The procedure, and a short program to demonstrate then follow.

```
10 FOR n=0 TO 359 STEP 10
     ellipse, n/5+30,20,n
   NEXT n
20 CLS
   FOR n=3 TO 8
     Polygon  n,30,,(n-2)*35,88
   NEXT n

500  DEF PROC ELLIPSE m,n,a,x,y.s
       DEFAULT a=0,x=128,y=88,s=30
       POLY s,m,n,x,y,a
     END PROC

510  DEF PROC POLYGON p,m,a,x,y
       DEFAULT a=0,x=128,y=88
       POLY p,m,,y,a
     END PROC

520  DEF PROC p,m,n,x,y,a
       LOCAL ca,sa,t,ct,st
       DEFAUL n=m,x=128,y=88,a=0
       LET a=a/180*PI
       LET ca=COSE(a),sa =SINE(a)
       PLOT x+m*ca,y+m*sa
       FOR t=0 TO 2*PI STEP 2*PI/p
         LET  ct=COSE(t),st=SINE(t)
         DRAW TO x+m*ca*ct-n*sa*st,y+m*sa*ct+n*ca*st
       NEXT t
       DRAW TO x+m*ca,y+m*sa
  END PROC
```

****

PROC md - improved Microdrive handling.

This contribution was sent in by A.J. Herstel, Holland. We have had a number of submissions on this theme, but this one has some really ingenious features. I have added line numbers to aid explanation, but the whole procedure was originally One line. The original replaced all uses of "I" with a variable to save memory, but I have altered this to aid clarity. Here is a modified version of Mr. Herstal's syntax description:

NEW COMMANDS

```
md <value>,<name>
md < 0 = SAVE              >,<>
md <more than 0 = RESAVE   >,<>
md < 1 = DEFAULT           >,<>
md <>,<"" = DEFAULT>
```

If you type Just:

```
md
```

you will see at the bottom of the screen:

```
RESAVE 1;"
```

You can complete the name, omitting the trailing quotes if you like, and the program named will be ERASEd from drive 1 before the current version is SAVEd and VERIFYed. If you had typed:

```
md 0
```

YOU would have seen:

```
SAVE 1; "
```

which gives SAVE and VERIFY without ERASE. You can alter the drive number by editing what appears at the bottom of the screen. LINE can be added, as a keyword or in lower-case letters, e.g.

```
RESAVE 1;"box" LINE 10
```

In a program you might use:

```
md ,"1;"" box"
```

```
LET f$="1;""box""code 65000,20": LET q=35: md q,f$
```

Because you have supplied a name, the procedure does not pause to let you type one. Note that the only significance of the first parameter is whether it is zero or non-zero.

```
1  DEF PROC md c,f$
     LOCAL p,q
     DEFAULT c=1,f =""
2    DO WHILE NOT LEN f$
       LET f$="1;"""
       EDIT "RESAVE " ((c=0)*2+1 TO ) ;  LINE f$
     LOOP UNTIL 1
4    LET p=INSTRING(1,f$,"""")+1,q=INSTRING(p,f$,"""")
     DO WHILE NOT q
       LET f$=f$+"""",q=LEN f$
     LOOP UNTIL 1
5    PRINT #1;AT 0,1;(p-3 TO q)
6    DO WHILE c
       PRINT #1; " ERASE ";
       KEYIN " ERASE "+f$(p-3 TO q)
7    LOOP UNTIL 1
     LET q=LEN f$-(LEN f$-q)*(INSTRING(q,f$," LINE ") OR
     INSTRING(q,f$,"line"))
     PRINT #1; " SAVE ";
     KEYIN " SAVE "+f$
     PRINT #1; " VERIFY "
     KEYIN " VERIFY "+f$( TO q)
8    INPUT ""
   END PROC
```

Comments: It took me a while to understand what was going on! Line 2 is equivalent to IF f$="" THEN.... The rest of the DO-LOOP allows the user to supply f$. LOOP UNTIL 1 means "never LOOP" - in other words, the LOOP's only purpose is to act as a marker, which can be jumped past by the DO WHILE, if f$ is not a null string.This makes the whole loop equivalent to:

```
IF f$="" then
   (statement)
END IF
```

This allows termination of IF-equivalents without use of a line number. The same structure is used to add a trailing quote IF one wasn’t provided (line 4), and to do an ERASE, IF the first parameter was non-zero (line 6)

Line 3 uses an interesting method of printing SAVE or RESAVE in the lower part of the screen. The multiple quote marks in lines 4 could be confusing. Quote marks inside strings have to be entered twice; the INSTRING functions are just looking for a single quote. Line 8 is used to clear the lower screen, without waiting for input.

Mr. Herstel also came up with the idea of renumbering the procedure as line zero, but this gave him errors, because he also renumber the original line zero as line 2 Don't do this! The original line , zero must be kept as line zero and it must be the first line in the program. Beta Basic executes the USR statement in that line by the equivalent of GOTO line 0, statement 23, after some Microdrive operations.

****

READERS LETTERS

Dear Andy,

Can you advise me over a BB SORT problem? I have dimensioned an array A$(60,2O) and have, so far, allocated 52 names only to this array pending the "arrival" of some more. SORT results in the printing of 8 blank spaces and then the names...I am presuming that an unfilled array must sort the blanks into the 1st position but can you suggest a way of avoiding this?

Charles Buszard, Chorleywood, Harts.

SORT considers a string of spaces to be as good a string as any other, and since the code for space comes before that for "a" or "A", the blanks come first. One answer is to use SORT A$(1 TO 52) which sorts only the first 52 strings . More flexibly, you could have followed the initial DIM statement with a FOR-NEXT loop setting every string to "zzzz". This would ensure every "unused" string was sorted to the end of the array.

****

Dear Dr. Wright,

Some time ago, I thought I would have a go at drawing a map of the world on the screen, and by using your ROLL make it revolve. Unfortunately, for some reason I now forget, I drew the world map in halves (i.e. SAVEing one half with a separate SAVE for the other half). The snag is that I cannot MERGE them into one. If I LOAD the two halves separately, then the second half obliterates the first. Can you suggest a way out, please?

The arrangement of the Spectrum's screen memory means that partial SAVES are only sensible if you save the top, middle or bottom thirds of the screen. (These start at 16384, 13432 and 20480 respectively, and are each 2048 bytes long. You have to use SAVE "name" CODE start,length - not SCREEN$.) You don't say whether your halves are top and bottom or left and right, but that doesn't matter for the method I propose. LOAD your first half, then use GET to save that part of the screen to a string - something like:

```
GET a$,0,175,16,22   (GET a$,x,y,width,length)
```

would be appropriate for the left half of the screen. Now LOAD the second half of your map - the first half will be erased by this, but you can use:

```
PLOT 0,175,a$
```

to restore it from your stored data. How SAVE it as one piece!

I have also been trying to put together a function (procedure) which will select (or find) the highest and lowest numbers in an unsorted list without first using SORT, and one to pick out the first and last in an ordered list.

John Davies, Cardiff

Let's do it for an ordered list first. If the array is full, the problem is easy, since the highest and lowest numbers come first and last, and we can use the LENGTH function to find out how many elements there are. If the array is only part used, you have a problem related in some ways to 11r. Buszard's (see previous page) except that unused elements in a numeric array are zero and get sorted to the bottom of the array, not the top. Now to "mark" an array element as unused? After all, it could really be zero! Me can use an unlikely value, such as -1e38, which shows that the element does not contain valid data and also ensures that it will be at the end of a sorted array. So let's set up some data for our procedure to chew on:

```
  10   DIM a(20)
  20   FOR n=1 TO 20
         LET a(n)=-1e38
       NEXT n
       REM initially all marked "invalid"
  30   FOR n=1 TO 10
         LET a(n)=RNDM(100)-40
       NEXT n
       REM put in 10 real numbers
  40   SORT a()
       REM sort them
  50   FOR n==1 TO 20
         PRINT a(n)
       NEXT n
       REM show the whole array
  60   ends a(),top,bot
       REM find max and min in a() and put results in top and bot
  70   PRINT
       PRINT "top=";top, "bot=";bot

  90   DEF PROC ends REF a(), REF hi, REF lo
  100    LOCAL el
  110    LET hi=a(1)
  120    LET eI=LENGTH(1,"a()")
  130    DO UNTIL a(el)>-1e38 OR el=1
  140      LET el=e1-1
  150    LOOP
  160    LET lo=a(el)
  170  END PROC
```

below is a procedure which finds the highest and lowest numbers in an unsorted list. It starts by assigning hi to have a very low value, correcting this as it looks through the array; low is treated in a similar way. "Unused" elements are not catered for - someone else can do that!

```
  10   DIM x(10)
  20   FOR n=1 TO 10
         LET  x(n)=RNDM (100 )
       NEXT n
       REM unsorted array set up
30   maxmin x(),a,b
     REM find max and min in x() and put results in a and b
40   PRINT "maximum= ";a
50   PRINT "minimum=";b
80   PRINT
     FOR n=1 TO 10
       PRINT x (n) ; " ;
     NEXT n
```

```
100  DEF PROC maxmin REF a(), REF hi, REF lo
110     LET hi=-1a38,lo=1e38
120     LOCAL n
130     FOR n=1 TO LENGTH(1,"a()")
140        IF a(n)>hi THEN LET hi=a(n)
150        IF a(n)<lo THEN LET  lo=a(n)
160     NEXT n
170  END PROC
```

****

Dear Andy,

Thank you very much for the newsletter No. 1 and for your personal answer to my question about the difference between UNTIL at the beginning or at the end of the DO LOOP.

(I told him that DO..... LOOP UNTIL will always execute the lines between DO and LOOP at least once, but that DO UNTIL . . . LOOP allows the possibility of not executing those lines even once - Ed.)

I am a 33 year old math's & computer teacher, living on a kibbutz near the sea of Galilee. Beta Basic is fantastic - it is like having bought a new computer. But don't think that this letter will be only praise.

(There followed a list of errors in Newsletter No, 1, which I covered on page 1 of this issue.)

As for the Beta Basic program itself, one improvement I would like to sea is in LIST DATA and LIST VAL$. If one of the string characters includes an unprintable character an error is printed and the rest of the string variables cannot be seen. I suggest that if the code is smaller than 32 a question mark, is printed instead, or that you warn users about this in the manual, so that a programmer using GET (a screen area) or MEMORY$ will use one of the last letters of the alphabet for this purpose!

Good point - I just didn't think of it - readers, consider yourselves warned until I can change the program!

If DEFAULT=t then LOAD 1;"name" will load from drive 1. But if DEFAULT=m then how the hell (*tsk tsk! - Ed.*) do I load from tape without having to write DEFAULT=t? I tried LOAD 0;"name" but it did not help.

You can't. I did not expect anyone with Microdrives to use tape very often - and if they did, I would expect the to use DEFAULT=t and type LOAD 1;"name" to use the drive. Using a "drive number" of zero to force tape to be used is an idea - but I'd have to take account of zero meaning "station 0" if DEFAULT=n.

. . . REF a$ 1000 TO 2000 would be helpful.. why doesn't REF work keywords? e.g. REF TRACE if I forgot where I included the bug tracing command! ... Is it possible to have two sets of small characters? Is there a system variable that points to 51291?

Daniel Bon-Sefer, Israel

I did consider allowing a range of line numbers to be specified for REF and ALTER. Hell, everything takes memory and programming time - wait for a new version REF does work with keywords - it will look for any string - try REF "TRACE", with TRACE being a keyword. (This can be entered as graphic-T.) You could have several small character sets. DPOKE 52406 with the address of the start of the small set, minus 112.

****

Dear Sir,

Although listings on the screen are indented neatly, on my printer any statements which carry over to the second line are nut indented. Why is this?

(A number of users)

When output goes to the screen, Beta Basic knows the line length and the current print position, and can control indentation. When a printer is being used, the problem is that different printers use different methods of storing that information. A solution would be to provide control of the line length in Beta Basic over-riding any interface, but I did not of this until too late.

****

Dear Sir,

RENUM does nut check LIST syntax i.e. LIST 10 TO 40 is renumbered correctly but LIST FORMAT, LIST DATA and LIST DEF are reported as incorrect syntax (*you mean, "Failed at" - Ed*) on renumbering. Could you also please tell me how I might transfer the program to EPROM as this will free more memory?

Clive Seaden, Hants.

RENUM was written before LIST FORMAT etc. and I forgot to change it to recognise them. You can just ignore the error messages that result. The question about EPROM’s is sometimes asked in almost exactly this way, to my surprise. I expected anyone with the facilities to program their own EPROM’s to realise that life isn't that simple! If you know how to SAVE a program onto EPROM and LOAD it back, fine, you should be able to do this with Beta Basic as easily as with a program you wrote yourself. But when you talk about freeing memory, you are implying a system that is able to live in a shadow EPROM, at a different address (requiring thousands of changes to the program) which can switch itself off temporarily while using the ordinary (requiring some electronics, which don't interfere with interface 1, and hundreds of program changes). Not easy - if you have to ask, you can't do it!

****

Dear Andy,

I'd like to point out that the binary to decimal conversion procedure in the first issue can be replaced by a simple function:

```
10 DEF FN z(b$)=VAL (“BIN”+b$)
```

Which is generally more convenient than a PROCedure to use.

BLUSH!

A list of Beta Basic system variables would be very much appreciated, and the ability to link a window to a particular stream e.g. OPEN #5,”w”;3 even more useful, OPEN #0,"w";7 which would allow input from any window.

Greg Harewood, 14 and a big bit.

The system variables will be documented in a future issue. I thought about linking windows to streams, but as you can PRINT or LIST to a window quite easily, I don't see much point for output.Input would be more valuable - we have received a lot of "INPUT AT" simulating contributions.

****
Dear Dr. Wright,

Your contributor aged 70, who claims difficulty in reading the red manual, is not alone. I am 78 end taught myself Basic when I retired. I write simple for a local business and for my grandchildren. The latter like Beta Basic as it puts them on a level with their school BBC computers. My version of Beta Basic is No. 10 - have there been any worthwhile updates since then? I have considered whether to go for a SPECTRUM 128 as my present one is rather an early model. I doubt the advantages but I would be influenced by a suitable BB program from yourself. What is the present position?

D.D. White, Clwyd.

Sorry about the manual. You have version 10 of BB 3.0 because PEEK 47272 is 10. As I write, there is a version 21, but this is not as bad as it sounds, since unlike the case with programs on chips, even minor buys are worth correcting quite soon. Here's a list of versions and what was corrected:

7 or less - Several problems - suggest you exchange your tape.
8 LOAD from tape with a non-zero CSIZE gave an error.
ON ERROR printed a number at the bottom of the screen.
9 SAVE *"m";1"name" (but not SAVE 1;"name") caused problems in procedures.
10 LISTing wasn't possible in WINDOWS only a few character-across.
11 REF was confused by ink, paper, inverse, or flash control codes with a zero parameter - POKE 48842,13 corrects.
12 LIST DEF KEY looked odd with certain def keys.
13 LLIST DATA had a problem with printing looping line numbers for FOR-NEXT loops. LIST DATA was fine!
14 A moderately exotic bug afflicted procedures. POKE 48591,205: POKE 48592,186: POKE 48593,24: POKE 52894,205: POKE 52895,186: POKE 52896,24 to fix.
15 The WINDOW example on page 66 of the manual was "messy".
16 Left-justification of procedure names was "messy".
17 Too exotic to explain easily.
18 REF sometimes found the same thing repeatedly. POKE 48354,86: POKE 48355,200: POKE 48742,86: POKE 48743,200: POKE 48937,86 : POKE 48938,200 will correct this.
19 SAVE *"m"1;"name" (but not SAVE 1;"name") sometimes caused Trouble.
20 Curtain control codes could turn off the LISTing of new keywords via the "t" channel on Interface 1, version 2.
21 Fix of the bug mentioned on page 1.

The vast majority of tapes sent out have be been version 9 or later If you wish, you can exchange your tape for the most recent version. He will pay the return postage.

I have spent some time getting to know the Spectrum 128. It has some nice features - lots of memory, a better display, improved sound and a RAMdisc. But I must say, considering the 17K of extra ROM space available, and the human and financial resources available to the manufacturers, the improvements to the Basic are pathetic. Most of the space is used in pointless repetition of the code in the original ROM. This might change - the ROM contents are not final yet - but I don't expect much from Betasoft's point of view, I suppose this is ideal, but I can't help being disappointed.

With the support of the software producers, the machine can do well - that extra memory (three times as much free) is important! But I would wait for the price to fall before buying. Beta Basic could do a lot with the 128 machine Then more requests we get, the sooner a version able to use the extra memory will be ready.(BB works fine in 48k mode.) However, the longer itto write the better it is likely to be!

****

Dear Dr. Wright,

Is there any way to implement CSIZE with LLIST on the ZX printer?

W.P. Trowbridge, Gateshead

My first impulse was "NO", use COPY, but this would be unlikely to give a perfect result with a long program. I have found a way to list the program one line at a time to the screen, followed by partial copy. This gives an automated "program COPY". It looks rather complicated (there is more on the next page) but it illustrates some points that may be of interest even to those without a ZX-type printer.

```
1  DEF PROC slist st,end
     DEFULT st=6,end=9999
     CLS
     LET x=DPEEK(23635)
     DO
       LET x=x+DPEEK (x+2)+4
       LET lnum=PEEK x*256+PEEK (x+1)
     LOOP UNTIL lnum>=st
     DO UNTIL lnum>end
       LIST lnum-1 TO lnum
       countlines rows
       partcopy rows
       LET lnum=PEEK x*256+PEEK (x+1)
       CLS
     LOOP
   END PROC
```

The procedure SLIST (screen LIST) uses the PROG system variable and the internal line information to find the first line number after st (start line for listing). This is done by the first DO-LOOP. The lines are then listed on the screen by LIST lnum-1 TO lnum. This may give problems if line numbers are in steps of one, but the more obvious LIST lnum TO lnum will give every line a ">" cursor. The second DO-LOOP lists each line separately to the screen, and copies it to the printer, up to the specified end line or the end of the program. Two sub-procedures are used: PROC countlines looks at the screen to find out how many screen lines the program line occupies, and PROC partcopy then sets up and uses a short machine code program to COPY that number of lines. (The code is in an unusual location - the calculator's memory area - which is easily corrupted, but quite usable in this case.)

```
  2  DEF PROC countlines REF row
       LOCAL col
       LET row=0,col=3
       DO
         IF SCREEN (row,col)<>" " " THEN
           LET row=row+1,col=3
         ELSE
           LET col=col+1
  3    LOOP UNTIL col>15
     END PROC

  4  DEF PROC partcopy rows
       POKE 23698,243
       POKE 20699,6
       POKE 23700,row*8
       POKE 23701,195
       POKE 23702,175
       POKE 20703,14
       RANDOMIZE USR 20699
     END PROC
```

Some examples: SLIST or SLIST 100 or SLIST 50,250
Only lines from line 6 onwards are SLISTable.

Several users have asked for some means of getting a text copy of the screen in CSIZE 4,8 onto a full-size printer. This would require a SCREEN$ that worked with that CSIZE. That would to quite easy; a version that worked with all CSIZEs would be much harder.

****

Dear Sirs,

Often, when a procedure is calla from within a program, report Q, "Parameter error" is received, even when no parameters are being passed. I enclose a listing ... it fails with 'Nonsense in Basic”. Is there a problem with the software?

I.D. Charles, Shrewsbury

The listing looked fine so I typed it in, and it worked without error. So what was the problem? The first 1 line was:

```
     1000 DEF PROC total
```

However, when I replaced it with:

```
1000 DEF PROC total(space)
```

the problem was encountered. This is not very user-friendly, and it will be changed. In the meantime, avoid trailing spaces on PROC names.

****

Dear Dr. Wright,

When I tried this:

```
10   LET a$=STRING$(7000,"a")
20   LET b$=STRING$(100,"b")
3O   JOIN a$ TO b$
```

It worked but it took over 6 minutes! This has given me a response time problem with a particular program where I wish to insert smaller strings into larger ones - the insertion could be anywhere therefore the requirement to JOIN large strings to smaller ones arises.

T.J. Vernon, Cambs.

The JOIN strings facility is a special case of the JOIN arrays command - strings are treated as one-dimensional arrays of LEN elements, each one character long. Because of the possible need to "stretch" an array element, and to keep temporary memory usage down, characters are added one-at-time to the destination array, and deleted from the source. In the process, large amounts of data have to be shuffled up and down, and doing this 7000 times is slow even in machine code. Incidentally, as someone (can't find the letter!) pointed out, if you use BREAK during this process, the variables area will be left in a corrupted state.

Your last sentence confuses me; you can JOIN a short string to a long one at any point. Your example will work much faster (6 seconds) if you change line 30 to:

```
30   JOIN b$ TO a$(1)
```

This JOINS the shorter string to the front of the longer, and only 100 characters have to be moved. Instead of (1) you could use any numeric expression.

(Mr. Vernon also reported "a rather nasty little bug" - if you LET a$="" (the null string) then EDIT a$ causes an irrecoverable crash. Be warned!)

****

LAST WORD

I still have lots of worthy contributions lying about, so the next issue should be ready "real soon now" as they say in the trade. Your opinions, queries and contributions are always welcome. I especially like contributions that are well explained, and not too long.

Scanned, Typed, OCR-ed, and PDF by

Steve Parry-Thomas 30th July 2004.

This PDF was created to preserve this
Newsletter for the future.

For all ZX Spectrum, Beta Basic
And www.worldofspectrum.org users

(PDF for Michael & Joshua)